Tabular Data Generation: Can We Fool XGBoost ?

El-Hacen Zein
hacen3064@gmail.com

Tanguy Urvoy
tanguy.urvoy@orange.com

Abstract

If by 'realistic' we mean indistinguishable from (fresh) real data, generating realistic synthetic tabular data is far from being a trivial task. We present here a series of experiments showing that strong classifiers like XGBoost are able to distinguish state-of-the-art synthetic data from fresh real data almost perfectly on several tabular datasets. By studying the important features of these classifiers, we remark that mixed-type (continuous/discrete) and ill-distributed numerical columns are the ones which are the less faithfully reconstituted. We hence propose and experiment a series of automated reversible column-wise encoders which improve the realism of the generators.

1 Introduction

Tabular data is the most common form of data in industry. It is ubiquitous in many key applications such as medical diagnosis, anomaly detection, predictive analytic, click-through rate prediction, user recommendation, or customer churn prediction [36, 5, 9]. In all these domains, the ability to generate realistic data has several applications such as privacy [52], testing [50], data imputation [13], oversampling [19] or explainability [26]. The flexibility of neural networks and their potential ability to 'understand' tabular data through the pre-training/fine-tuning paradigm are appealing research avenues with several applications [21, 4, 17]. However, despite recent efforts to design adapted architectures [3, 28], tabular data still poses a huge challenge for neural networks on predictive tasks where boosted decision trees like XGBoost [15] remain the best option both for performances and training time (see [23, 9, 49, 25, 20]).

A key difficulty of tabular data is the *heterogeneity* of features. A table mostly contains categorical features (sex, occupation,...) which are reputed unordered and discrete, and numerical features (age, capital-gain...) which are frequently assumed to be ordered and continuous. But things are often more complex: Categorical values can exhibit some ordering or semantic structure, they are often highly unbalanced with scarce, missing, or even erroneous values. Numerical values can behave like mixtures of discrete, truncated or continuous distributions. They are often ill-distributed with long tails, extreme outliers, and various scales. All these numerical oddities, well described in [56] and [11], can have a deleterious effect on the models that are trained with gradient descent.

Despite recent efforts to deploy adapted neural architectures [5, 9], a practitioner willing nowadays to work with neural networks and tabular data is bound to handcraft a problem-specific encoding/decoding procedure for each table column [49]. Recent works on tabular data suggest that proper encoding methods rather than architecture innovations can significantly boost performances of discriminative and generative models [9, 24, 55].

Most of these issues are shared between predictive and generative models. But generative tasks also induce additional difficulties. Contrary to purely predictive models, the encoding procedures for generative models also have to be reversible. Another difficulty is evaluation. Evaluating generative models is known to be a delicate issue in general [54], but for tabular data, unlike in computer vision where a visual sanity check can be performed, the visual checking of a single generated tabular row, when not meaningless, requires a deep knowledge of the problem.

36th Table Representation Learning workshop (NeurIPS 2022).

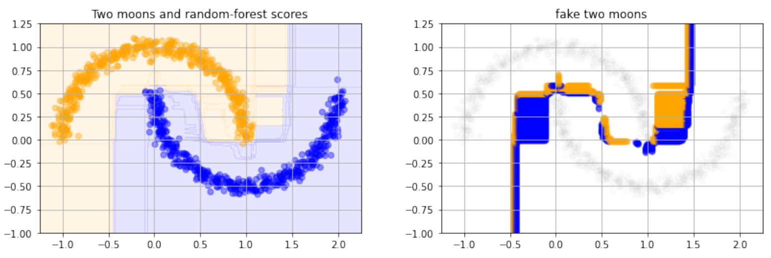

Figure 1: An illustrative example showing that single-target ML-efficacy is not a sufficient measure. On the left hand side the original 'two-moons' dataset with the score map of a Random-Forest classifier that obtains a perfect test AUC of 99.9%. On the right-hand side a very unrealistic dataset that frames the decision boundary. A new classifier trained on this 'framing' data obtains a perfect test AUC as well: decision framing is the simplest way to maximize ML-efficacy.

Our contributions are as follows: In section 2 we first highlight with a toy example that single-target ML-efficacy, a popular metric in the literature on deep tabular data generation, is not robust to inconsistencies between real and fake samples. We hence argue, following [37], that the *Classifier Two-Sample Test* (C2ST) is more relevant, and way more challenging than ML-efficacy to assess the realism of synthetic tabular data.

We then show in Section3 that strong classifiers are able to distinguish state-of-the-art synthetic data from real data almost perfectly on several tabular datasets. We highlight with a post-hoc analysis of these classifiers that the poor performances of the generative models we tested are mainly dues to mixed-type and ill-distributed numerical features. In sections 4 and 5 we finally propose three – data-driven – encoding schemes for numerical features and show that some of them allow models to generate fake samples with consistently improved C2ST scores.

2 Classifier two-sample test as a realism measure for tabular data

Several measures can be employed to evaluate the performance of generative models for tabular data [46]. A classical sanity check consists of visually comparing the histograms for each column [16, 45, 14]. This comparison can also be automated by using two-sample statistics like Kolmogorov-Smirnov [6, 37], but averaging single-column metrics does not take into account the inter-dependencies between features. Other metrics proposed in [46] average multi-columns aggregates to circumvent this problem, but the aggregate's choice is rather arbitrary.

A popular metric that takes the dependencies into account is *machine learning efficacy* (ML-efficacy). Driven by privacy applications, this metric relates to the (test) performance gap between predictive models trained respectively on real and artificial data [16, 12, 6, 45, 55, 56, 34]. This metric can be single-target or averaged on multiple target columns. Most publications however use single-target ML-efficacy as it is less costly to compute.

But, as a matter of fact, the best way to maximize ML-efficacy is not to generate realistic data. It is sufficient to train a classifier on original data and generate samples that properly frame its decision boundaries. We illustrate this fact with a toy example in Figure 1.

We propose to evaluate the tabular generative models with a less frequently used metric called *Classifier 2-sample test* (C2ST) or *Detection test* in SDV [46]. This metric, well studied in [37], consists in quantifying the ability of a binary classifier to discriminate fresh real data from fake data. The C2ST methodology proceeds as follow: We first split the original dataset into a train set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$. We train the generative model on $\mathcal{D}_{train}$ and use it to generate an artificial test set $\mathcal{D}_{syn}$ with $|D_{syn}| = |D_{test}|$. We then construct the labeled dataset $\mathcal{T} = \mathcal{D}_{test} \times \{\text{real}\} \sqcup \mathcal{D}_{syn} \times \{\text{fake}\}$ that we randomly split into a train set $\mathcal{T}_{train}$ and a test set $\mathcal{T}_{test}$. We finally train a 'discriminant' classifier on $\mathcal{T}_{train}$ to estimates the probability for an instance to be fake. The performance of this discriminant classifier on $\mathcal{T}_{test}$ is our C2ST statistic.

In our experiments we assess this performance with the *Area Under the ROC Curve* (AUC): an AUC of 0.5 means that synthetic data is indistinguishable from real data, while an AUC of 1 means that synthetic instances are easily spotted by the classifier. Note that, contrary to the discriminator model in *Generative Adversarial Networks* (GANs) [22], the discriminant classifier we use is trained and tested from scratch on fresh data. We can experience overfitted GANs with poor C2ST scores, but where the internal discriminator is fooled by its generator. As developed in the next section, we found through extensive experiments that recent GAN-based or VAE-based generative models that could reach high ML-efficacy scores were easily discriminated from the real data through an XGBoost-based two sample test.

3 Synthetic tabular data is easily spotted by XGBoost

Most of the numerous proposed approaches for tabular data generation provide both architecture innovations and encoding/decoding innovations. Among the non-neural methods, *copulas* [33, 43, 48, 46] are popular because they allows us to model the marginals and the features inter-dependencies separately. There have also been some recent attempts to combine copulas with graphical models [41, 7], and neural networks [53, 30, 32, 1, 46].

Among the deep generative models, several variants of GANs have been tested and adapted in order to cope with tabular data [2, 16, 12, 42, 45, 31, 14]. A Few Variational Auto-Encoders (VAE) have been tested as well [55, 38, 26], and some recent works also applied normalizing FLOWS on tabular data [32, 35]. Among the proposed models, conditional generators like **CTGAN** and **TVAE** [55], CWGAN [19], and CTAB-GAN [56] are the most flexible for data imputation or oversampling, and their ability to learn-by-sampling is an efficient counter-measure against categorical imbalance training issues.

3.1 Classifier-based two-sample test experiment

As a first experiment, we evaluate the realism of three representative tabular data generators in terms of C2ST. We chose to test **CTGAN**, **TVAE**, and GAUSSIAN COPULAS generators because these models are known to achieve state-of-the-art performance in term of ML-efficiency and because they are publicly available with – automated – features encoding/decoding schemes as part of the SDV project [48, 46]. Another interesting improvement proposed with **CTGAN** and **TVAE** is the use of Gaussian mixtures (GM) for mode-specific normalization of numerical features. Some interesting improvements have also been proposed in [56], but the proposed solutions mostly rely on handcrafted features engineering.

For this experiment we used eight real world datasets from the benchmark provided in [55]. We added to this benchmark, a few artificial datasets provided in scikit-learn [47]. We modified the SDV evaluation procedure in order to add C2ST score with stronger classifiers[1]: Gradient Boosting, Random Forest, Multi-Layer Perceptron (MLP) from Scikit-learn [47] and XGboost [15]. Each generative model was trained on a 70% split using default hyper-parameters. The C2ST's classifiers were trained and evaluated using a 3-fold cross validation with their default hyper-parameters.

The results are summarized in Figure 2. We found that all models: **CTGAN**, **TVAE** and GAUSSIAN COPULAS obtain poor C2ST scores, especially on real-world datasets where all classifiers are able to discern fake data from real data with an AUC that is greater than 0.80. Moreover, as expected, the XGboost classifier is outperforming the other classifiers, detecting fake data with an AUC greater than 0.92 for all real datasets.

Our first experiments revealed two interesting result: *(i)* Although better than the ones generated by GAUSSIAN COPULAS, the fake examples generated by **CTGAN** and **TVAE** still contain patterns that allowed all classifiers to discriminate them from real examples. *(ii)* Achieving high ML-efficacy scores is not sufficient to affirm that the fake samples follow the same distribution as the real data.

3.2 Post-hoc analysis of the discriminating classifiers

In order to explain the poor scores obtained on Figure 2, we retrieve two standard feature-importance measures from the discriminating classifiers [44]: the impurity-based importance and the permutation-based importance. In Figure 3, we compare, for three datasets, the importance of numerical features against categorical (multi-class and binary) features, revealing that numerical features have a much higher impact than categorical features.

[1]See SDMetrics patch on github.com/sdv-dev/SDMetrics/pull/235

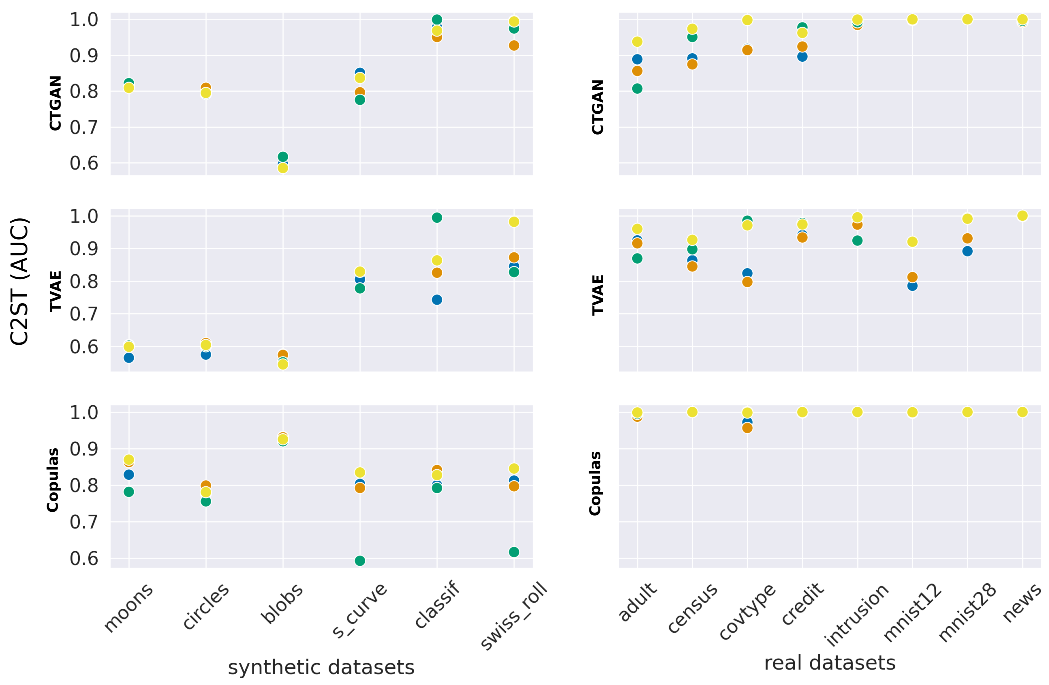

Figure 2: C2ST scores using respectively **CTGAN** (first row); **TVAE** (second row); and GAUSSIAN COPULAS generators (last row) with their original hyper-parameters and encodings on several datasets. We tested different classifiers: scikitlearn's gradient boosting in blue, scikitlearn's random forest in orange, Multi-Layers Perceptron (MLP) with 100 hidden neurons in green, and XGBoost in yellow. Recall that an AUC of 1.0 means that the classifier detects all fake instances while an AUC of 0.5 means perfect indistinguishability.

Our results show that the conditional generative models are doing well with the sparsity of categorical features, but still fail at efficiently capturing the complexity of multivariate numerical values. As mentioned in the introduction, the behaviour of numerical features are often complex with multiple scales, multiple modes, long-tails, and mixed-types with continuous distributions, discrete events and truncation effects. These numerical oddities are illustrated in Figure 4.

4 Improving numerical encoders

As we have seen previously, the samples generated by **CTGAN** and **TVAE** can perfectly be distinguished from real data (Figure 2), and we believe that this is mainly due to the lack of a proper numerical feature representation. Our goal in this work is not to propose another network architecture, but rather to explore new approaches for numerical features encoding and study their impact on the realism of the generated samples.

We use *one-hot encoding* for categorical features. The – derivable – reverse transform for this encoding is to return the *Gumbel-softmax* [29]. Various strategies such as hierarchy, embeddings or hashing, have been proposed to deal with large dimensions (see [27] for a survey), but several were designed for input data and are not reversible, hence not suitable for data generation.

For numerical features the most basic transform is normalization. But, as pointed-out in [51], in presence of multi-modal distributions, normalization does not prevent a GAN to collapse into a single mode (even on a 2D dataset). Our baseline will be the approach of [55], called *mode-specific normalization*, which consists in fitting a *Variational Gaussian Mixture* model [8] on each numeric feature. We will call this encoder VGM encoder. We categorize the encodings we propose in two types: *feature binning* and *marginal-density estimation*.

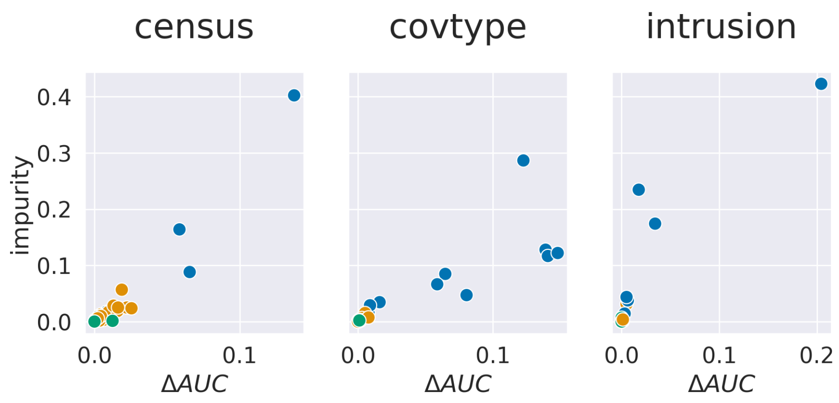

Figure 3: A scatter plot with two feature importance metrics retrieved from a classifier trained to discriminate samples generated by **CTGAN** from real data. The y-axis shows the impurity-based feature importance. The x-axis shows the ΔAUC permutation importance. Each dot is a table column with numerical columns in blue, multiClass columns in orange and binary columns in green.

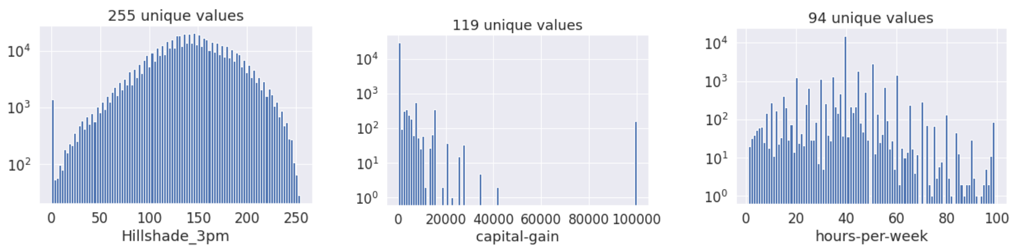

Figure 4: Examples of ill-distributed numerical features with Diracs, long-tails and semi-discrete behaviours (with log-scale). On the left-hand side `Hillshade_3pm` (from covtype) with a heavy peak in zero meaning *"no-measure"*. In the middle `capital-gain` with an extreme outlier peak meaning *"more than* 100000*"*, and on the right-hand side `hours-per-week` with several isolated peaks, one for each work contract.

4.1 Feature binning

Feature binning is a long-existing discretization technique where the value range is partitioned into disjoint intervals called bins, and where every scalar value is represented by its corresponding bin index. This lossy encoding is known to improve Naive Bayes classifiers [18, 10]. Recently, [24] studied the effect of binning on predictive neural networks and proposed a lossless binning encoder, called *Piecewise Linear Encoding* (PLE), where the edges of the bins are used to construct piece-wise linear representations of the original scalar values. We propose here two adaptations of *feature binning* for generative models.

Let X be a numerical feature, and let $b_0, b_1, \dots, b_m$ be its bin edges determined trough equal-frequency or another method. Given a value $x \in X$, the bin index i_x is the index verifying $b_{i_x} \leq x < b_{i_x+1}$. We propose two encoding schemes:

Piecewise Linear Encoding (PLE) This method, directly inspired from [24], takes an input value $x \in X$ and returns the concatenation of a one-hot vector $v = (v_0, \dots, v_m)$ for the bin index i_x and a scalar value α representing the min-max normalization of x inside its bin. Formally, $\mathrm{ple}(x) = v \oplus \alpha$ with $v_i = \mathbb{1}_{b_i \leq x < b_{i+1}}$ and $\alpha = \frac{x - b_{i_x}}{b_{i_x+1} - b_{i_x}}$.

The scalar α being in the interval $[0, 1]$, it can be generated using a sigmoid function. The one-hot representation v is generated with a Gumbel-softmax [29]. The inverse transform is obtained by $x = (1 - \alpha) \cdot b_{i_x} + \alpha \cdot b_{i_x+1}$.

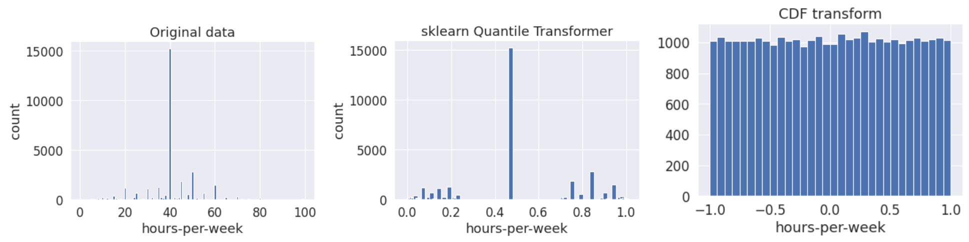

Figure 5: Contrary to the standard deterministic quantile transform which preserves the discrete behaviours, the randomized CDF "dequantizes" the original distribution into a smooth uniform. Although randomized, this transformation does not lose information.

Prototype Encoding (PTP) The prototype encoding method takes inspiration from prototypical networks [40]. The idea consists of viewing the input value $x \in X$ as a weighted average of the edges/prototypes values $b_0, \dots, b_m$. The input value x is hence encoded into a vector of positive weights $\mathrm{ptp}(x) = (w_0, \dots, w_m)$ such that $\sum_{i=0}^{m} w_i \cdot b_i = x$. This representation is not unique. In order to keep the weights as sparse as possible we define the forward encoding by:

$$w_i = \begin{cases} 1-\alpha & \text{when} \quad i = i_x, \\ \alpha & \text{when} \quad i = i_x + 1, \\ 0 & \text{otherwise} \end{cases}$$

Where i_x and α are defined as for the PLE encoder. The PTP representation of a scalar can be generated with a softmax function. The inverse transform is a simple scalar product: $\mathrm{ptp}(x) \cdot (b_0, \dots, b_m)$.

4.2 Marginal-density estimation

Following the same approach as with copulas, we fit for each numerical feature a model that maps its distribution to a uniform one, hence leaving only for the generative model to learn the joint distribution between the uniform marginals. This approach has been already explored, for example in [38] using marginal VAEs and in [32] using univariate normalizing flows. However, fitting and tuning a neural network for each feature is computationally expensive, especially with high-dimensional datasets. We propose here the *cumulative-distribution-function* (CDF) encoder, which relies directly on the raw *empirical* CDF to perform a smooth encoding.

Formally let $f_x = \{x_1, \dots x_n\}$ be the train values of a numerical feature. We define the two following functions:

$$F(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{x_i < x} \quad \text{and} \quad D(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{x_i = x}$$

Note that $F(x)$ (resp. $D(x)$) is an approximation of $\mathbb{P}(X < x)$ (resp. $\mathbb{P}(X = x)$). The forward encoding of a value $x \in \mathbb{R}$ is $F(x)$ or a uniform sample from $[F(x), F(x) + D(x)]$ if $x \in f_x$. As described on Figure 5, this encoding transforms any peaked numerical feature into a *"dequantized"* continuous representation that is guaranteed to be uniformly distributed in $[0, 1]$. It hence differs from the deterministic `QuantileTransformer` from scikitlearn which always returns $F(x)$. The reverse transform is $F^{-1}(u) = \max\{x \mid F(x) \leq u\}$.

4.3 Hybrid encoder

We have seen throughout our experiments that both the binning-based encoders and the density-based encoder achieve competitive results compared to the VGM encoder (Figure 6). The CDF encoder makes it extremely easy to capture the marginal distribution but misses the redundant encoding that the binning method provides. To take advantage of both methods we designed PLE_CDF: an hybridization of the CDF encoder and the PLE encoder. It consists of first, applying a CDF encoder to transform a numerical feature into a uniform distribution, then applying the PLE encoder on the resulting transformation.

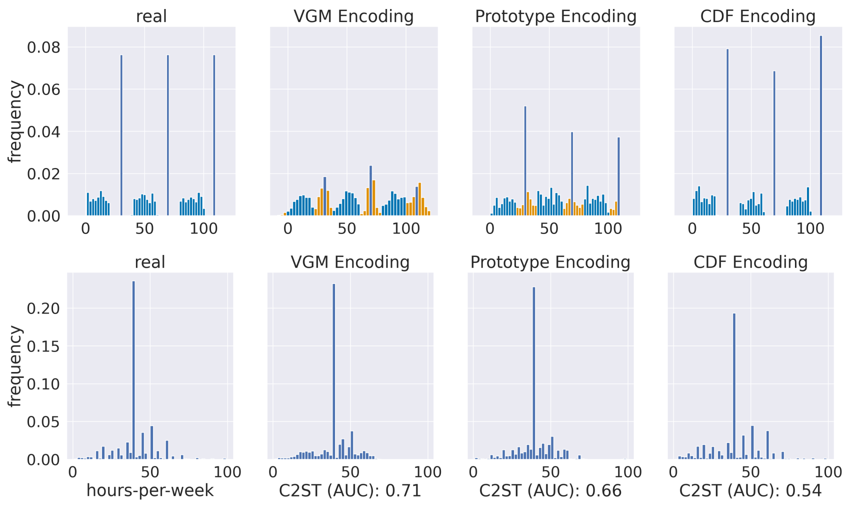

Figure 6: Behavior of a **CTGAN** model in presence of mixed-type distributions (first row) and multi-modal distributions (second row). On the far left side is the target distribution. The other three figures show the histograms of **CTGAN**-generated samples using respectively a VGM encoder, a PTP encoder, and a CDF encoder. The task in the first row is to generate a mix of three uniform distributions and three Dirac impulses. Erroneous samples are highlighted in orange. Note the inability of Gaussian mixtures and prototypes to model Diracs impulses. In the second row the real data correspond to the hours-per-week in adult dataset. To facilitate the comparison between the models we computed a univariate C2ST score, which shows that the samples synthesized using the CDF encoder and the PTP encoder are harder to discriminate from real data.

5 Experiment

In this section we empirically evaluate the impact of the techniques discussed in section 4 on **CTGAN** and **TVAE**, and compare them with the VGM encoder [55].

We use C2ST as detailed in Section2. To make the task harder we use the strongest classifier in our preliminary experiments, i.e, The XGBoost classifier. We also use ML-efficacy and two additional metrics for privacy and overfitting: *Distance to Closest Records* (DCR) and *Nearest Neighbor Distance Ratio* (NNDR) [39]. These two metrics are here to prevent from a generative model that would just sample rows from training data, even if, as mentioned in [37], a sufficiently large set of generated samples would reveal such memorization to the two-sample test. All reported measures are the averaged results of a 3-fold cross validation. For the ML-efficacy scores we report the average measure (AUC for binary classification and Accuracy for multiclass classification) of all used classifiers.

For each model (**CTGAN** and **TVAE**) and each dataset we compared the performances using the baseline Gaussian mode-specific normalisation (denoted VGM) against its performance using the techniques introduced in section 4 (PLE, PTP, CDF and PLE_CDF). Each model was trained for 300 epochs using the default hyperparameters in the original paper [55].

The results for four real-world datasets are summarized in Tables 1, 2, 3 and 4. The first row, named **Identity**, gives the performance of a perfect generator, i.e., a model that would generate exactly the test set. The main takeaways of this experiment are:

- For 3 out of the 4 datasets we used, our encoders achieved, on each model, significant improvement in terms of C2ST scores over the existing VGM encoder, with the PLE_CDF encoder consistently achieving the best scores.
- All models passed the over-fitting DCR test (only one occurrence of p-value lower than 0.1)
- All models have an NNDR higher than the test set (Identity).
- Our encoders do not seem to have any significant impact on the ML-efficacy. We can safely say that all encoders have similar ML-efficacy scores.

		C2ST↓	DCR↑	NNDR↑	ML-efficacy↑
Identity		0.51±0.00	1.00±0.00	0.33±0.00	0.79±0.01
TVAE	VGM	0.97±0.00	0.18±0.20	0.38±0.03	0.77±0.02
	PLE	0.97±0.00	0.09±0.04	0.47±0.01	0.76±0.02
	CDF	0.82±0.01	0.30±0.26	0.37±0.00	0.76±0.03
	PLE_CDF	0.91±0.02	0.09±0.08	0.37±0.02	0.72±0.03
CTGAN	VGM	0.99±0.00	0.17±0.03	0.47±0.01	0.76±0.03
	PTP	0.90±0.01	0.29±0.13	0.48±0.00	0.77±0.03
	PLE	0.81±0.01	0.22±0.05	0.46±0.01	**0.78±0.01**
	CDF	0.85±0.01	0.29±0.05	**0.52±0.02**	0.76±0.03
	PLE_CDF	**0.76±0.01**	**0.31±0.15**	0.48±0.00	0.77±0.02

Table 1: Results on *Adult* dataset.

		C2ST↓	DCR↑	NNDR↑	ML-efficacy↑
Identity		0.50±0.00	1.00±0.00	0.15±0.02	0.71±0.07
TVAE	VGM	0.91±0.01	**0.64±0.11**	0.29±0.04	0.66±0.05
	PLE	0.93±0.00	0.53±0.23	0.33±0.02	0.65±0.06
	CDF	0.89±0.01	0.62±0.23	0.33±0.02	0.63±0.10
	PLE_CDF	0.86±0.01	0.51±0.18	0.31±0.03	0.68±0.07
CTGAN	VGM	0.99±0.00	0.46±0.13	0.56±0.01	0.72±0.06
	PTP	0.88±0.00	0.36±0.07	0.54±0.04	0.70±0.07
	PLE	0.89±0.01	0.38±0.04	0.53±0.03	0.70±0.07
	CDF	0.94±0.00	0.59±0.15	**0.58±0.04**	**0.73±0.05**
	PLE_CDF	**0.85±0.01**	0.46±0.07	0.56±0.03	0.72±0.07

Table 2: Results on *Census* dataset.

		C2ST↓	DCR↑	NNDR↑	ML-efficacy↑
Identity		0.50±0.00	1.00±0.00	0.38±0.00	0.90±0.05
TVAE	VGM	0.97±0.00	0.40±0.03	0.69±0.02	0.72±0.04
	PLE	0.99±0.00	0.57±0.24	**0.76±0.01**	0.70±0.01
	CDF	**0.91±0.00**	0.51±0.15	0.71±0.00	**0.72±0.05**
	PLE_CDF	0.94±0.02	0.37±0.20	0.71±0.01	0.71±0.03
CTGAN	VGM	0.98±0.00	0.65±0.24	0.71±0.01	0.67±0.08
	PTP	0.95±0.00	**0.70±0.13**	0.69±0.01	0.67±0.08
	PLE	0.98±0.01	0.43±0.21	0.68±0.00	0.66±0.08
	CDF	0.95±0.00	0.59±0.26	0.67±0.00	0.68±0.08
	PLE_CDF	0.93±0.01	0.61±0.17	0.69±0.00	0.67±0.08

Table 3: Results on *covtype* dataset.

		C2ST↓	DCR↑	NNDR↑	ML-efficacy↑
Identity		0.50±0.00	1.00±0.00	0.00±0.00	0.87±0.02
TVAE	VGM	0.91±0.01	0.40±0.13	0.09±0.00	0.50±0.00
	PLE	0.94±0.01	0.48±0.20	0.08±0.02	0.56±0.02
	CDF	0.97±0.00	0.49±0.07	**0.77±0.02**	0.70±0.12
	PLE_CDF	0.98±0.01	**0.60±0.17**	0.03±0.03	0.60±0.05
CTGAN	VGM	**0.85±0.01**	0.53±0.13	0.35±0.02	0.92±0.01
	PTP	0.93±0.00	0.36±0.07	0.10±0.02	0.92±0.01
	PLE	0.91±0.01	0.58±0.12	0.23±0.03	0.93±0.01
	CDF	0.94±0.00	0.26±0.24	0.68±0.01	0.92±0.03
	PLE_CDF	0.88±0.01	0.44±0.18	0.47±0.04	**0.93±0.01**

Table 4: Results on *credit* dataset.

6 Conclusion

We showed that on tabular data, a strong classifier like XGBoost is perfectly able to distinguish the real samples from the fake ones synthesized by state-of-the-art generative models. We also highlighted that ill-distributed numerical columns are the less faithfully reconstituted. We proposed three new automatic encoding schemes that can transform ill-distributed numerical features into regular representations that are suitable for neural networks. We empirically showed that these methods improve the synthetic samples quality without deteriorating the ML-efficacy and the privacy metrics. Furthermore theses encodings are simple, computationally efficient and easily adaptable to any neural network model (predictive or generative). However despite this improvement, with AUC scores higher than 0.80, our synthesized samples are still easily spotted by XGBoost. Developing models and encodings able to lower this score a step further is an exciting challenge.

7 Acknowledgement

We would like to thank the TRL reviewers for their constructive feedback. This work was performed using HPC/AI resources from GENCI-IDRIS (Grant 2022-AD011012220R1).

References

[1] CopulaGAN Model — SDV 0.16.0 documentation, 2022. URL `https://sdv.dev/SDV/user_guides/single_table/copulagan.html`.

[2] M. Alzantot, S. Chakraborty, and M. Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193. IEEE, 2017.

[3] S. O. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning, 2019. URL `https://arxiv.org/abs/1908.07442`.

[4] S. Ö. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

[5] I. Ashrapov. Review of GANs for tabular data, June 2020. URL `https://towardsdatascience.com/review-of-gans-for-tabular-data-a30a2199342`.

[6] M. K. Baowaly, C.-C. Lin, C.-L. Liu, and K.-T. Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association : JAMIA*, 26(3):228—241, March 2019. ISSN 1067-5027. doi: 10.1093/jamia/ocy142. URL `https://europepmc.org/articles/PMC7647178`.

[7] F. Benali, D. Bodénès, N. Labroche, and C. de Runz. Mtcopula: Synthetic complex data generation using copula. In *23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*, pages 51–60, 2021.

[8] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. Deep Neural Networks and Tabular Data: A Survey. Technical Report arXiv:2110.01889, arXiv, Feb. 2022. URL `http://arxiv.org/abs/2110.01889`. arXiv:2110.01889 [cs] type: article.

[10] M. Boullé. Modl: a bayes optimal discretization method for continuous attributes. *Machine learning*, 65(1):131–165, 2006.

[11] M. Boullé and V. Zelaia. Floating-point histograms for exploratory analysis of large scale real-world data sets. Technical report, Orange-labs, 2022.

[12] R. Camino, C. Hammerschmidt, and R. State. Generating multi-categorical samples with generative adversarial networks. 2018. doi: 10.48550/ARXIV.1807.01202. URL `https://arxiv.org/abs/1807.01202`.

[13] R. Camino, C. Hammerschmidt, and R. State. Working with deep generative models and tabular data imputation. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020. URL `https://openreview.net/forum?id=R4w3PTkCD4`.

[14] H. Chen, S. Jajodia, J. Liu, N. Park, V. Sokolov, and V. S. Subrahmanian. Faketables: Using gans to generate functional dependency preserving tables with bounded real data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2074–2080. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/287. URL `https://doi.org/10.24963/ijcai.2019/287`.

[15] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco California USA, Aug. 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `https://dl.acm.org/doi/10.1145/2939672.2939785`.

[16] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun. Generating multi-label discrete patient records using generative adversarial networks. In F. Doshi-Velez, J. Fackler, D. Kale, R. Ranganath, B. Wallace, and J. Wiens, editors, *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 286–305. PMLR, 18–19 Aug 2017. URL `https://proceedings.mlr.press/v68/choi17a.html`.

[17] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. TURL: table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319, Nov. 2020. ISSN 2150-8097. doi: 10.14778/3430915.3430921. URL `https://dl.acm.org/doi/10.14778/3430915.3430921`.

[18] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. J. Russell, editors, *ICML*, pages 194–202. Morgan Kaufmann, 1995. ISBN 1-55860-377-8. URL `http://dblp.uni-trier.de/db/conf/icml/icml1995.html#DoughertyKS95`.

[19] J. Engelmann and S. Lessmann. Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, 174:114582, 2021.

[20] S. A. Fayaz, M. Zaman, S. Kaul, and M. A. Butt. Is deep learning on tabular data enough? an assessment. *International Journal of Advanced Computer Science and Applications*, 13(4), 2022.

[21] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189, Lille, France, July 2015. JMLR.org.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[23] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[24] Y. Gorishniy, I. Rubachev, and A. Babenko. On Embeddings for Numerical Features in Tabular Deep Learning. *arXiv e-prints*, art. arXiv:2203.05556, Mar. 2022.

[25] L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

[26] V. Guyomard, F. Fessant, T. Bouadi, and T. Guyet. Post-hoc counterfactual generation with supervised autoencoder. In M. e. a. Kamp, editor, *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 105–114, Cham, 2021. Springer International Publishing. ISBN 978-3-030-93736-2.

[27] J. Hancock and T. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7, 04 2020. doi: 10.1186/s40537-020-00305-w.

[28] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv e-prints*, pages arXiv–2012, 2020.

[29] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[30] T. Janke, M. Ghanmi, and F. Steinke. Implicit generative copulas. *Advances in Neural Information Processing Systems*, 34:26028–26039, 2021.

[31] J. Jordon, J. Yoon, and M. Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

[32] S. Kamthe, S. Assefa, and M. Deisenroth. Copula flows for synthetic data generation. *arXiv preprint arXiv:2101.00598*, 2021.

[33] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[34] J. Lee, J. Hyeong, J. Jeon, N. Park, and J. Cho. Invertible tabular gans: Killing two birds with one stone for tabular data synthesis. *Advances in Neural Information Processing Systems*, 34: 4263–4273, 2021.

[35] J. Lee, M. Kim, Y. Jeong, and Y. Ro. Differentially private normalizing flows for synthetic tabular data generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (7):7345–7353, Jun. 2022. doi: 10.1609/aaai.v36i7.20697. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20697`.

[36] D. Libes, D. Lechevalier, and S. Jain. Issues in synthetic data generation for advanced manufacturing. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1746–1754. IEEE, 2017.

[37] D. Lopez-Paz and M. Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, page 14, Vancouver, 2017.

[38] C. Ma, S. Tschiatschek, R. Turner, J. M. Hernández-Lobato, and C. Zhang. Vaem: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems*, 33:11237–11247, 2020.

[39] J. M. Mateo-Sanz, F. Sebé, and J. Domingo-Ferrer. Outlier protection in continuous microdata masking. In *International Workshop on Privacy in Statistical Databases*, pages 201–215. Springer, 2004.

[40] P. Mettes, E. van der Pol, and C. Snoek. Hyperspherical prototype networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/02a32ad2669e6fe298e607fe7cc0e1a0-Paper.pdf`.

[41] D. Meyer, T. Nagler, and R. J. Hogan. Copula-based synthetic data augmentation for machine-learning emulators. *Geoscientific Model Development*, 14(8):5205–5215, 2021.

[42] A. Mottini, A. Lheritier, and R. Acuna-Agost. Airline passenger name record generation using generative adversarial networks. *CoRR*, abs/1807.06657, 2018. URL `http://arxiv.org/abs/1807.06657`.

[43] R. B. Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

[44] S. Nembrini, I. R. König, and M. N. Wright. The revival of the Gini importance? *Bioinformatics*, 34(21):3711–3718, 05 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty373. URL `https://doi.org/10.1093/bioinformatics/bty373`.

[45] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, jun 2018. doi: 10.14778/3231751.3231757. URL `https://doi.org/10.14778%2F3231751.3231757`.

[46] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016. doi: 10.1109/DSAA.2016.49.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[48] C. Sala, M. A. Campo, J. D. P. Cañellas, and K. Xiao. The Synthetic Data Vault. Put synthetic data to work!, 2021. URL `https://sdv.dev/`.

[49] R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, May 2022. ISSN 15662535. doi: 10.1016/j.inffus.2021.11.011. URL `https://linkinghub.elsevier.com/retrieve/pii/S1566253521002360`.

[50] G. Soltana, M. Sabetzadeh, and L. C. Briand. Synthetic data generation for statistical testing. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 872–882. IEEE, 2017.

[51] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *Advances in neural information processing systems*, 30, 2017.

[52] T. Stadler, B. Oprisanu, and C. Troncoso. Synthetic data–anonymisation groundhog day. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1451–1468, 2022.

[53] N. Tagasovska, D. Ackerer, and T. Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. *Advances in neural information processing systems*, 32, 2019.

[54] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. URL `http://arxiv.org/abs/1511.01844`.

[55] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling Tabular data using Conditional GAN. *Advances in Neural Information Processing Systems*, page 11, 2019.

[56] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen. CTAB-GAN: Effective Table Data Synthesizing. *Asian Conference on Machine Learning*, page 16, 2021.